CONTENTS:

Program 1 in C:

```
/*
 * PHYS319 Lab3 Timing example in C
 *
 * Written by Ryan Wicks
 * 16 January 2012
 *
 * This program is a C version of the assembly program that formed part of lab 2.
 * This is not the best way to implement timing, or to organize your code.
 * It is simply one way.
 *
 * This will almost certainly not give exactly the same timing as the assembly
 * program from lab 2, and the output assembly will also be very different, even
 * though the task is similar.
 */

#include <msp430.h>

void main(void) {
      volatile unsigned int count;  //You must declare your variables in C
  // notice the label volatile. What happens if you remove this label?

      WDTCTL = WDTPW + WDTHOLD;     //Stop WDT
      P1DIR = 0x41;                 //Set P1 output direction
      P1OUT = 0x01;                 //Set the output

      while (1){                    //Loop forever
            count = 60000;
            while(count != 0) {
                  count--;          //decrement
            }
            P1OUT = P1OUT ^ 0x41;   //bitwise xor the output with 0x41
      }
}
```

Program 2 in C:

```
/*
 * PHYS319 Lab 3 Interrupt Example in C
 *
 * Written by Ryan Wicks
 * 16 Jan 2012
 *
 * This program is a C version of the assembly program that formed part of
 * lab 2.
 *
 *
 */
#include  <msp430.h>

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;      // Stop watchdog timer
  P1DIR = 0xF7;               //C does not have a convenient way of
                              //representing numbers in binary; use hex instead
  P1OUT = 0x49;
  P1REN = 0x08;                 //enable resistor
  P1IE = 0x08;                     //Enable input at P1.3 as an interrupt

  _BIS_SR (LPM4_bits + GIE);  //Turn on interrupts and go into the lowest
                              //power mode (the program stops here)
            //Notice the strange format of the function, it is an "intrinsic"
            //ie. not part of C; it is specific to this chipset
}

// Port 1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void PORT1_ISR(void)
{                             //code goes here
  P1OUT ^= 0x41;           // toggle the LEDS
  P1IFG &= ~0x08;          // Clear P1.3 IFG. If you don't, it just happens again.
}
```

ADC demo:

```
//*******************************************************************************
//  MSP430G2x31 Demo - ADC10, Sample A1, AVcc Ref, Set P1.0 if > 0.75*AVcc
//
//  Description: A single sample is made on A1 with reference to AVcc.
//  Software sets ADC10SC to start sample and conversion - ADC10SC
//  automatically cleared at EOC. ADC10 internal oscillator times sample (16x)
//  and conversion.
//
//               MSP430G2x31
//            -----------------
//         /|\|              XIN|-
//          | |                 |
//          --|RST          XOUT|
//            |                 |
//            |                 |
//            |                 |
//            |                 |
//            |                 |
// input  >---|P1.1/A1      P1.0|--> red Led onboard BIT0
//            |                 |
//            |             P1.2|--> yellow Led
//            |             P1.6|--> green Led onboard BIT6
//
//
//  D. Dang
//  Texas Instruments Inc.
//*******************************************************************************
#include "msp430.h"

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  ADC10CTL0 = ADC10SHT_2 + ADC10ON;          // ADC10ON
  ADC10CTL1 = INCH_1;                       // input A1
  ADC10AE0 |= 0x02;                         // PA.1 ADC option select
  P1DIR |= 0x01 ;                            // Set P1.0 to output direction

  for (;;)
  {
    ADC10CTL0 |= ENC + ADC10SC;             // Sampling and conversion start
    while (ADC10CTL1 &ADC10BUSY);          // ADC10BUSY?
    if (ADC10MEM < 0x2FF)
      P1OUT &= ~0x01;                       // Clear P1.0 LED off
    else
      P1OUT |= 0x01;                        // Set P1.0 LED on

    unsigned i;
    for (i = 0xFFFF; i > 0; i--);           // Delay
  }
}
//*******************************************************************************
```

Table 3-3. Source/Destination Operand Addressing Modes

As/Ad	Addressing Mode	Syntax	Description
00/0	Register mode	Rn	Register contents are operand
01/1	Indexed mode	X(Rn)	(Rn + X) points to the operand. X is stored in the next word.
01/1	Symbolic mode	ADDR	(PC + X) points to the operand. X is stored in the next word. Indexed mode X(PC) is used.
01/1	Absolute mode	&ADDR	The word following the instruction contains the absolute address. X is stored in the next word. Indexed mode X(SR) is used.
10/-	Indirect register mode	@Rn	Rn is used as a pointer to the operand.
11/-	Indirect autoincrement	@Rn+	Rn is used as a pointer to the operand. Rn is incremented afterwards by 1 for .B instructions and by 2 for .W instructions.
11/-	Immediate mode	#N	The word following the instruction contains the immediate constant N. Indirect autoincrement mode @PC+ is used.

The seven addressing modes are explained in detail in the following sections. Most of the examples show the same addressing mode for the source and destination, but any valid combination of source and destination addressing modes is possible in an instruction.

NOTE: Use of Labels *EDE, TONI, TOM, and LEO*

Throughout MSP430 documentation EDE, TONI, TOM, and LEO are used as generic labels. They are only labels. They have no special meaning.

3.4.6.4 AND

AND[.W] Source AND destination

AND.B Source AND destination

Syntax

```
AND src,dst  or  AND.W src,dst
AND.B src,dst
```

Operation

```
src .AND. dst → dst
```

Description The source operand and the destination operand are logically ANDed. The result is placed into the destination.

Status Bits

N: Set if result MSB is set, reset if not set

Z: Set if result is zero, reset otherwise

C: Set if result is not zero, reset otherwise (= .NOT. Zero)

V: Reset

Mode Bits OSCOFF, CPUOFF, and GIE are not affected.

Example The bits set in R5 are used as a mask (#0AA55h) for the word addressed by TOM. If the result is zero, a branch is taken to label TONI.

```
MOV     #0AA55h,R5   ; Load mask into register R5
AND     R5,TOM       ; mask word addressed by TOM with R5
JZ      TONI         ;
......               ; Result is not zero
;
;
;   or
;
;
AND     #0AA55h,TOM
JZ      TONI
```

Example The bits of mask #0A5h are logically ANDed with the low byte TOM. If the result is zero, a branch is taken to label TONI.

```
AND.B   #0A5h,TOM    ; mask Lowbyte TOM with 0A5h
JZ      TONI         ;
......               ; Result is not zero
```

3.4.6.25 JEQ, JZ

JEQ, JZ Jump if equal, jump if zero

Syntax

```
JEQ label
JZ label
```

Operation If Z = 1: PC + 2 offset → PC

If Z = 0: execute following instruction

Description The status register zero bit (Z) is tested. If it is set, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If Z is not set, the instruction following the jump is executed.

Status Bits Status bits are not affected.

Example Jump to address TONI if R7 contains zero.

```
TST   R7             ; Test R7
JZ    TONI           ; if zero: JUMP
```

Example Jump to address LEO if R6 is equal to the table contents.

```
CMP   R6,Table(R5)   ; Compare content of R6 with content of
                     ; MEM (table address + content of R5)
JEQ   LEO            ; Jump if both data are equal
......               ; No, data are not equal, continue here
```

Example Branch to LABEL if R5 is 0.

```
TST   R5
JZ    LABEL
......
```

3.4.6.28 JMP

JMP	Jump unconditionally
Syntax	`JMP label`
Operation	PC + 2 × offset → PC
Description	The 10-bit signed offset contained in the instruction LSBs is added to the program counter.
Status Bits	Status bits are not affected.
Hint	This one-word instruction replaces the BRANCH instruction in the range of –511 to +512 words relative to the current program counter.

3.4.6.31 JNE, JNZ

JNE Jump if not equal

JNZ Jump if not zero

Syntax

```
JNE label
JNZ label
```

Operation

If Z = 0: PC + 2 a offset → PC

If Z = 1: execute following instruction

Description The status register zero bit (Z) is tested. If it is reset, the 10-bit signed offset contained in the instruction LSBs is added to the program counter. If Z is set, the next instruction following the jump is executed.

Status Bits Status bits are not affected.

Example Jump to address TONI if R7 and R8 have different contents.

```
CMP   R7,R8   ; COMPARE R7 WITH R8
JNE   TONI    ; if different: jump
......        ; if equal, continue
```

3.4.6.32 MOV

MOV[.W] Move source to destination

MOV.B Move source to destination

Syntax

```
MOV src,dst  or  MOV.W src,dst
MOV.B src,dst
```

Operation src → dst

Description The source operand is moved to the destination.

The source operand is not affected. The previous contents of the destination are lost.

Status Bits Status bits are not affected.

Mode Bits OSCOFF, CPUOFF,and GIE are not affected.

Example The contents of table EDE (word data) are copied to table TOM. The length of the tables must be 020h locations.

```
       MOV   #EDE,R10                ; Prepare pointer
       MOV   #020h,R9                ; Prepare counter
Loop   MOV   @R10+,TOM-EDE-2(R10)    ; Use pointer in R10 for both tables
       DEC   R9                      ; Decrement counter
       JNZ   Loop                    ; Counter not 0, continue copying
       ......                        ; Copying completed
       ......
       ......
```

Example The contents of table EDE (byte data) are copied to table TOM. The length of the tables should be 020h locations

```
       MOV     #EDE,R10              ; Prepare pointer
       MOV     #020h,R9              ; Prepare counter
Loop   MOV.B   @R10+,TOM-EDE-1(R10)  ; Use pointer in R10 for
                                     ; both tables
       DEC   R9                      ; Decrement counter
       JNZ   Loop                    ; Counter not 0, continue
                                     ; copying
       ......                        ; Copying completed
       ......
       ......
```

3.4.6.51 XOR

XOR[.W]	Exclusive OR of source with destination
XOR.B	Exclusive OR of source with destination
Syntax	`XOR src,dst  or  XOR.W src,dst` `XOR.B src,dst`
Operation	src .XOR. dst → dst
Description	The source and destination operands are exclusive ORed. The result is placed into the destination. The source operand is not affected.
Status Bits	N: Set if result MSB is set, reset if not set Z: Set if result is zero, reset otherwise C: Set if result is not zero, reset otherwise (= .NOT. Zero) V: Set if both operands are negative
Mode Bits	OSCOFF, CPUOFF,and GIE are not affected.
Example	The bits set in R6 toggle the bits in the RAM word TONI. `XOR     R6,TONI  ; Toggle bits of word TONI on the bits set in R6`

Example	The bits set in R6 toggle the bits in the RAM byte TONI.
	`XOR.B   R6,TONI  ; Toggle bits of byte TONI on the bits set in`
	`                ; low byte of R6`

Example	Reset to 0 those bits in low byte of R7 that are different from bits in RAM byte EDE.
	`XOR.B   EDE,R7   ; Set different bit to "1s"`
	`INV.B   R7       ; Invert Lowbyte, Highbyte is 0h`

22.3.1 ADC10CTL0, ADC10 Control Register 0

15	14	13	12	11	10	9	8
SREFx			**ADC10SHTx**		**ADC10SR**	**REFOUT**	**REFBURST**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
MSC	**REF2_5V**	**REFON**	**ADC10ON**	**ADC10IE**	**ADC10IFG**	**ENC**	**ADC10SC**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

Can be modified only when ENC = 0

SREFx Bits 15-13 Select reference.

- 000 $V_{R+} = V_{CC}$ and $V_{R-} = V_{SS}$
- 001 $V_{R+} = V_{REF+}$ and $V_{R-} = V_{SS}$
- 010 $V_{R+} = V_{eREF+}$ and $V_{R-} = V_{SS}$. Devices with V_{eREF+} only.
- 011 V_{R+} = Buffered V_{eREF+} and $V_{R-} = V_{SS}$. Devices with V_{eREF+} pin only.
- 100 $V_{R+} = V_{CC}$ and $V_{R-} = V_{REF-}/ V_{eREF-}$. Devices with V_{eREF-} pin only.
- 101 $V_{R+} = V_{REF+}$ and $V_{R-} = V_{REF-}/ V_{eREF-}$. Devices with $V_{eREF+/-}$ pins only.
- 110 $V_{R+} = V_{eREF+}$ and $V_{R-} = V_{REF-}/ V_{eREF-}$. Devices with $V_{eREF+/-}$ pins only.
- 111 V_{R+} = Buffered V_{eREF+} and $V_{R-} = V_{REF-}/ V_{eREF-}$. Devices with $V_{eREF+/-}$ pins only.

ADC10SHTx Bits 12-11 ADC10 sample-and-hold time

- 00 4 × ADC10CLKs
- 01 8 × ADC10CLKs
- 10 16 × ADC10CLKs
- 11 64 × ADC10CLKs

ADC10SR Bit 10 ADC10 sampling rate. This bit selects the reference buffer drive capability for the maximum sampling rate. Setting ADC10SR reduces the current consumption of the reference buffer.

- 0 Reference buffer supports up to ~200 ksps
- 1 Reference buffer supports up to ~50 ksps

REFOUT Bit 9 Reference output

- 0 Reference output off
- 1 Reference output on. Devices with V_{eREF+} / V_{REF+} pin only.

REFBURST Bit 8 Reference burst.

- 0 Reference buffer on continuously
- 1 Reference buffer on only during sample-and-conversion

MSC Bit 7 Multiple sample and conversion. Valid only for sequence or repeated modes.

- 0 The sampling requires a rising edge of the SHI signal to trigger each sample-and-conversion.
- 1 The first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed

REF2_5V Bit 6 Reference-generator voltage. REFON must also be set.

- 0 1.5 V
- 1 2.5 V

REFON Bit 5 Reference generator on

- 0 Reference off
- 1 Reference on

ADC10ON Bit 4 ADC10 on

- 0 ADC10 off
- 1 ADC10 on

ADC10IE Bit 3 ADC10 interrupt enable

- 0 Interrupt disabled
- 1 Interrupt enabled

ADC10IFG	Bit 2	ADC10 interrupt flag. This bit is set if ADC10MEM is loaded with a conversion result. It is automatically reset when the interrupt request is accepted, or it may be reset by software. When using the DTC this flag is set when a block of transfers is completed. 0 No interrupt pending 1 Interrupt pending
ENC	Bit 1	Enable conversion 0 ADC10 disabled 1 ADC10 enabled
ADC10SC	Bit 0	Start conversion. Software-controlled sample-and-conversion start. ADC10SC and ENC may be set together with one instruction. ADC10SC is reset automatically. 0 No sample-and-conversion start 1 Start sample-and-conversion

22.3.2 ADC10CTL1, ADC10 Control Register 1

15	14	13	12	11	10	9	8
INCHx				**SHSx**		**ADC10DF**	**ISSH**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC10DIVx			**ADC10SSELx**		**CONSEQx**		**ADC10BUSY**
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r-0

Can be modified only when ENC = 0

INCHx Bits 15-12 Input channel select. These bits select the channel for a single-conversion or the highest channel for a sequence of conversions. Only available ADC channels should be selected. See device specific data sheet.

- 0000 A0
- 0001 A1
- 0010 A2
- 0011 A3
- 0100 A4
- 0101 A5
- 0110 A6
- 0111 A7
- 1000 V_{eREF+}
- 1001 V_{REF-}/V_{eREF-}
- 1010 Temperature sensor
- 1011 (V_{CC} - V_{SS}) / 2
- 1100 (V_{CC} - V_{SS}) / 2, A12 on MSP430F22xx devices
- 1101 (V_{CC} - V_{SS}) / 2, A13 on MSP430F22xx devices
- 1110 (V_{CC} - V_{SS}) / 2, A14 on MSP430F22xx devices
- 1111 (V_{CC} - V_{SS}) / 2, A15 on MSP430F22xx devices

SHSx Bits 11-10 Sample-and-hold source select.

- 00 ADC10SC bit
- 01 Timer_A.OUT1 (1)
- 10 Timer_A.OUT0 (1)
- 11 Timer_A.OUT2 (Timer_A.OUT1 on MSP430F20x0, MSP430G2x31, and MSP430G2x30 devices) (1)

ADC10DF Bit 9 ADC10 data format

- 0 Straight binary
- 1 2s complement

ISSH Bit 8 Invert signal sample-and-hold

- 0 The sample-input signal is not inverted.
- 1 The sample-input signal is inverted.

ADC10DIVx Bits 7-5 ADC10 clock divider

- 000 /1
- 001 /2
- 010 /3
- 011 /4
- 100 /5
- 101 /6
- 110 /7
- 111 /8

ADC10SSELx Bits 4-3 ADC10 clock source select

- 00 ADC10OSC
- 01 ACLK
- 10 MCLK
- 11 SMCLK

(1) Timer triggers are from Timer0_Ax if more than one timer module exists on the device.

CONSEQx	Bits 2-1	Conversion sequence mode select
	00	Single-channel-single-conversion
	01	Sequence-of-channels
	10	Repeat-single-channel
	11	Repeat-sequence-of-channels
ADC10BUSY	Bit 0	ADC10 busy. This bit indicates an active sample or conversion operation
	0	No operation is active.
	1	A sequence, sample, or conversion is active.

22.3.3 ADC10AE0, Analog (Input) Enable Control Register 0

7	6	5	4	3	2	1	0
ADC10AE0x							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

ADC10AE0x	Bits 7-0	ADC10 analog enable. These bits enable the corresponding pin for analog input. BIT0 corresponds to A0, BIT1 corresponds to A1, etc. The analog enable bit of not implemented channels should not be programmed to 1.
	0	Analog input disabled
	1	Analog input enabled

22.3.4 ADC10AE1, Analog (Input) Enable Control Register 1 (MSP430F22xx only)

7	6	5	4	3	2	1	0
ADC10AE1x				**Reserved**			
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

ADC10AE1x	Bits 7-4	ADC10 analog enable. These bits enable the corresponding pin for analog input. BIT4 corresponds to A12, BIT5 corresponds to A13, BIT6 corresponds to A14, and BIT7 corresponds to A15. The analog enable bit of not implemented channels should not be programmed to 1.
	0	Analog input disabled
	1	Analog input enabled
Reserved	Bits 3-0	Reserved

22.3.5 ADC10MEM, Conversion-Memory Register, Binary Format

15	14	13	12	11	10	9	8
0	0	0	0	0	0	**Conversion Results**	
r0	r0	r0	r0	r0	r0	r	r
7	6	5	4	3	2	1	0
Conversion Results							
r	r	r	r	r	r	r	r

Conversion Results	Bits 15-0	The 10-bit conversion results are right justified, straight-binary format. Bit 9 is the MSB. Bits 15-10 are always 0.

22.3.6 ADC10MEM, Conversion-Memory Register, 2s Complement Format

15	14	13	12	11	10	9	8
Conversion Results							
r	r	r	r	r	r	r	r
7	6	5	4	3	2	1	0
Conversion Results		0	0	0	0	0	0
r	r	r0	r0	r0	r0	r0	r0

Conversion Results	Bits 15-0	The 10-bit conversion results are left-justified, 2s complement format. Bit 15 is the MSB. Bits 5-0 are always 0.

22.3.7 ADC10DTC0, Data Transfer Control Register 0

7	6	5	4	3	2	1	0
Reserved				**ADC10TB**	**ADC10CT**	**ADC10B1**	**ADC10FETCH**
r0	r0	r0	r0	rw-(0)	rw-(0)	r-(0)	rw-(0)

Reserved	Bits 7-4	Reserved. Always read as 0.
ADC10TB	Bit 3	ADC10 two-block mode 0 One-block transfer mode 1 Two-block transfer mode
ADC10CT	Bit 2	ADC10 continuous transfer 0 Data transfer stops when one block (one-block mode) or two blocks (two-block mode) have completed. 1 Data is transferred continuously. DTC operation is stopped only if ADC10CT cleared, or ADC10SA is written to.
ADC10B1	Bit 1	ADC10 block one. This bit indicates for two-block mode which block is filled with ADC10 conversion results. ADC10B1 is valid only after ADC10IFG has been set the first time during DTC operation. ADC10TB must also be set. 0 Block 2 is filled 1 Block 1 is filled
ADC10FETCH	Bit 0	This bit should normally be reset.

22.3.8 ADC10DTC1, Data Transfer Control Register 1

7	6	5	4	3	2	1	0
DTC Transfers							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)

DTC Transfers	Bits 7-0	DTC transfers. These bits define the number of transfers in each block. 0 DTC is disabled 01h-0FFh Number of transfers per block

22.3.9 ADC10SA, Start Address Register for Data Transfer

15	14	13	12	11	10	9	8
ADC10SAx							
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)
7	6	5	4	3	2	1	0
ADC10SAx							0
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0

ADC10SAx	Bits 15-1	ADC10 start address. These bits are the start address for the DTC. A write to register ADC10SA is required to initiate DTC transfers.
Unused	Bit 0	Unused, Read only. Always read as 0.